NORTHERN ILLINOIS UNIVERSITY

Stacker: A Microworld for Algebra

A Thesis submitted to the

University Honors Program

in Partial Fulfillment of the

Requirements of the Baccalaureate Degree

With University Honors

Department of Mathematical Sciences

by

Melissa Elli Hauser

Dekalb, Illinois

May 1988

JOURNAL OF THESIS ABSTRACTS

THESIS SUBMISSION FORM

AUTHOR: Melissa Elli Hauser

THESIS TITLE: Stacker: A Microworld for Algebra

THESIS ADVISOR: Dr. Gary Martin

ADVISOR'S DEPT: Mathematical Sciences

DATE: May 1988

HONORS PROGRAM: NIU

NAME OF COLLEGE: Northern Illinois University

PAGE LENGTH: 17

BIBLIOGRAPHY (YES OR NO): Yes

ILLUSTRATED (YES OR NO): No

COPIES AVAILABLE (HARD COPY,MICROFILM,DISKETTE): Diskette

SUBJECT HEADINGS: (CHOOSE FIVE KEY WORDS) Mathematics Education, Computers, Education, Microworld, Teaching Algebra

ABSTRACT (100-200 words):

Research has been done which shows students have difficulty bridging the gap between arithmetic and basic algebra. Often, students who are able to perform symbolic manipulations still have an inadequate understanding of the underlying concepts involved. Further research has shown that emphasizing the link between arithmetic and algebra aids students in acquiring these concepts. A microworld is a type of computer program which allows exploration within the confines of an environment which models a conceptual framework. Research suggests that this kind of exploration is especially conducive to internalization of the framework being modeled. "Stacker: A Microworld for Algebra", pictorally models the processes involved in solving simple linear equations in one variable.

For Office Use:

THESIS NUMBER:

ABSTRACT

Research has been done which shows students have difficulty bridging the gap between arithmetic and basic algebra. Often, students who are able to perform symbolic manipulations still have an inadequate understanding of the underlying concepts involved. Further research has shown that emphasizing the link between arithmetic and algebra aids students in acquiring these concepts. A microworld is a type of computer program which allows exploration within the confines of an environment which models a conceptual framework. Research suggests that this kind of exploration is especially conducive to internalization of the framework being modeled. "Stacker: A Microworld for Algebra", pictorially models the processes involved in solving simple linear equations in one variable.

RATIONALE

In this section, we discuss the desireability of providing students with additional pre-algebraic experiences and the potential of computer generated microworlds to provide such experiences.

Definition

A microworld is an environment created to allow free exploration within some conceptual framework. A microworld consists of a set of objects, relationships among objects, and operations that manipulate the objects (Thompson, 1985). Some key characteristics are the presence of one or more clearly defined goals, the absence of negative reinforcement and the absence of a well defined procedure to follow to achieve the goal(s). A microworld is explored through utilization of a series of commands which allow the student/user to manipulate the elements of the microworld in accordance with the operations of the microworld. The end result is achievement of a previously specified goal, in a "game like" environment. The conditions and commands of a microworld parallel real life principles and processes of the conceptual framework being modeled. This facilitates

internalization of the framework being modeled and transfer to future learning situations. The rationale for this is eloquently expressed by Seymore Papert (1981), "... anything is easy if you can assimilate it to your collection of models. If you can't, anything can be painfully difficult."

Background Research

The teaching and learning of Algebra is an important issue in math education today. Researchers spend a good deal of time examining the reasons many students have difficulty truly understanding and retaining basic algebraic concepts. A few statistics will clearly illustrate the problem. In a study of 17 year olds with two years of algebra, only 64% of them could simplify an expression like $7 - 8(3 - y)$ and only 75% could solve an equation of the form $9 = 5x + 2$ for x (Carpenter, Corbitt, Kepner, Lindquist, and Reys 1981).

Research suggests that there is a break down in the understanding of mathematical concepts between arithmetic and algebra. This is due largely to the way in which they are presented. Arithmetic concepts are easily demonstrated using real objects and are typically introduced in this manner. Algebra, on the other hand, is typically presented in a highly abstract, symbolic format. Students who are unable to fully grasp the meaning of the concepts involved tend to memorize patterns of

manipulating symbols in an effort to "get by".

Research also suggests that emphasizing the link between arithmetic, which most students understand, and algebra can strengthen understanding of algebraic concepts. "Stacker" is based on techniques introduced by Herscovics and Kieran (1980) which have proved successful in bridging the gap between arithmetic and algebra. Herscovics and Kieran introduce equations as "arithmetic identities" such as the following

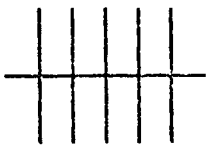
$$4 \times 2 + 2 = 5 \times 2.$$

The truth of an arithmetic identity is easily verified by a student who understands arithmetic, thus an intuitive feeling for equality is built. Variables are introduced as "blanks" or covered up numbers in an arithmetic identity. Students are given the task of determining the number which is covered up - - the number which makes the identity true. Students are able to cope with variables presented in this way; it forms a link between arithmetic and the symbolic representation of variables.

OVERVIEW OF STACKER

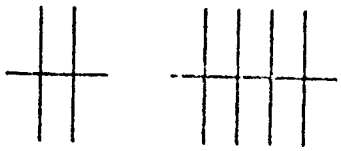
"Stacker" is a computer generated microworld which allows students to explore arithmetic identities and eventually progress to solving simple linear equations in one variable. The objects of this microworld are sticks tied together in bundles. The bundle below represents the number 5, and an "x" is represented by a box.

There are four commands or operations with which a user may manipulate bundles. They are BUNDLE, UNBUNDLE, STACK and UNSTACK. BUNDLE and UNBUNDLE model addition, STACK and UNSTACK model multiplication. Below are examples of each of the commands. Letters are used to refer to terms (i.e. "A" refers to the first term etc.).



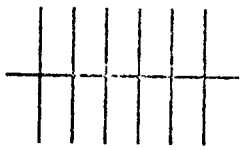
5

BUNDLE requires two inputs which correspond to the terms to be "bundled or added



2 + 4

BUNDLE A B

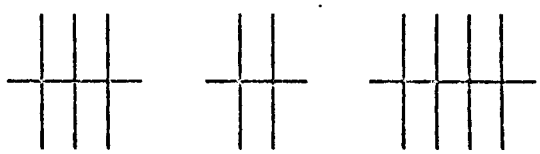


6

UNBUNDLE requires two inputs - the first is the term to be "unbundles the second, the size of one of the bundles to be created out of the specified term

5 + 4

UNBUNDLE A 3



3 + 2 + 4

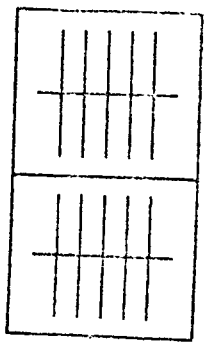
STACK requires 3 inputs - the first corresponds to the term to be stacked, the second and third correspond to the height of the stack and what is to be in each box of the stack

6

STACK A 3 2

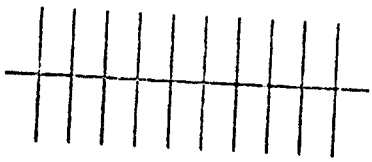
3(2)

UNSTACK requires only one input - the term to be unstacked



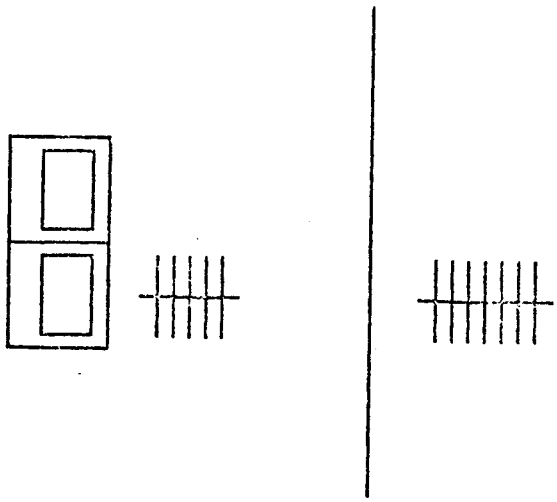
2(5)

UNSTACK A



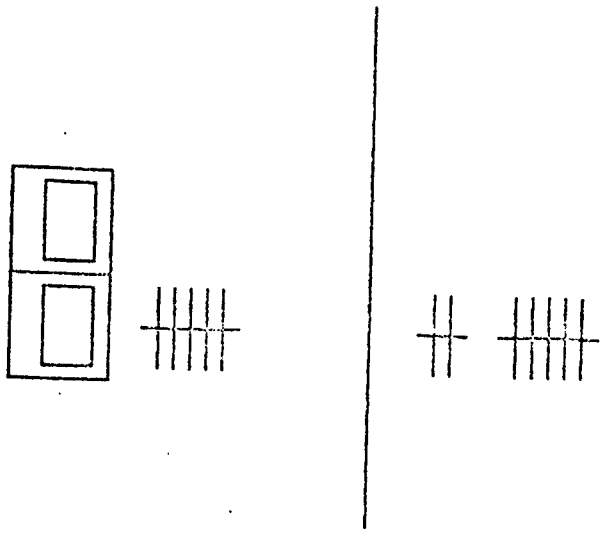
10

"Stacker" utilizes a goal format in which the user enters the command "GOAL n" to specify that he wishes to attempt goal number n. This command causes goal state n to be initiated and the user is free to persue the goal using the available commands. Below is an example of the graphic display of a goal with its symbolic representation underneath. The goal is to make the right side look like the left, thus discovering what belongs in the box (the value of x). The steps in reaching the goal are illustrated on the next page.



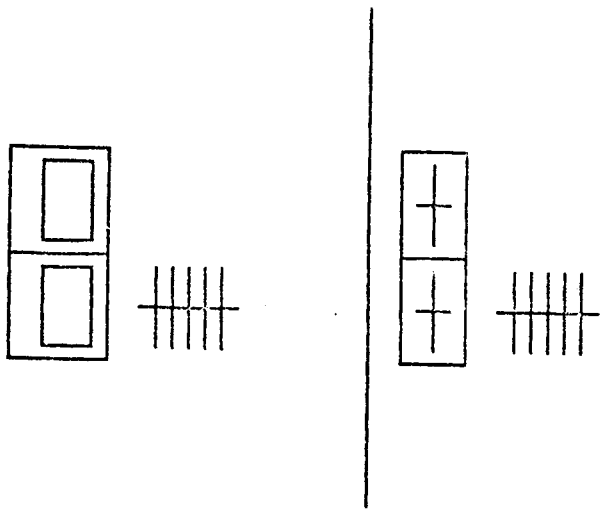
$$2X + 5 = 7$$

UNBUNDLE A 2 produces this



$$2X + 5 = 2 + 5$$

and STACK A 2 1 results in the following



$$2X + 5 = 2(1) + 5$$

we can now see that 2 sticks should go in the box

RECOMMENDATIONS FOR USE

A classroom teacher of a course such as pre-algebra or general math may find "Stacker" a useful building block in creating a solid foundation for the introduction of algebraic concepts. The key element which gives it this potential is the methodology used in this program. Although use of the computer itself is often viewed as a novel and highly motivational approach, it must not be allowed to overshadow the real objectives of the microworld.

The objectives of "Stacker" are as follows:

Objectives

1) to create an environment which fosters true understanding as opposed to blind manipulation of symbols

2) to build a solid arithmetic foundation for the introduction of equations and variables

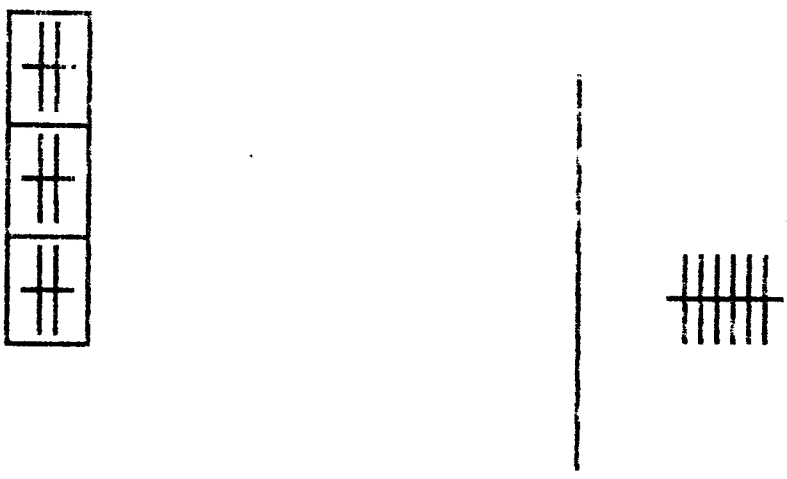
3) to guide students the to discovery of a pedogogically sound method of solving simple linear equations in one variable

Outline

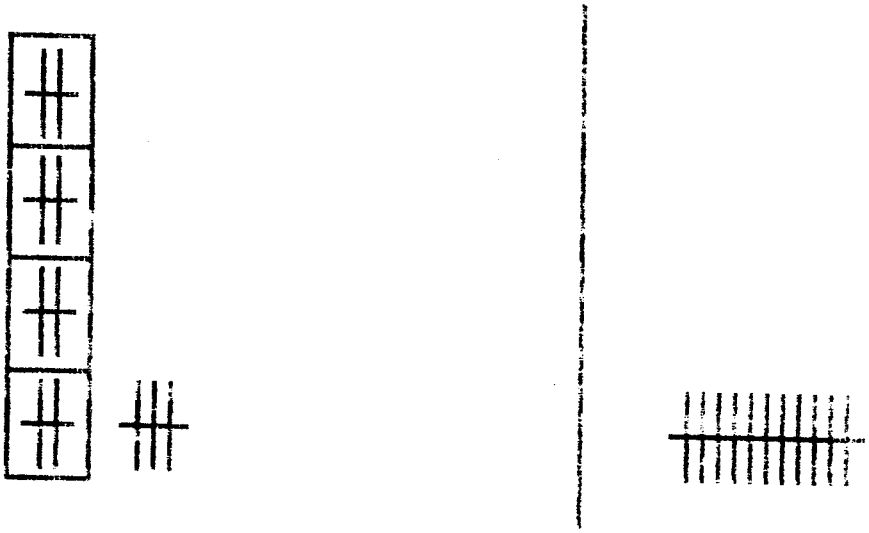
Below is a narrative of a sample set of goal states to achieve the objectives stated above. It should be noted that this sequence is intended as a sample outline which would require augmentation with additional goals for classroom use.

this goal is designed to allow the students to practice the UNBUNDLE command

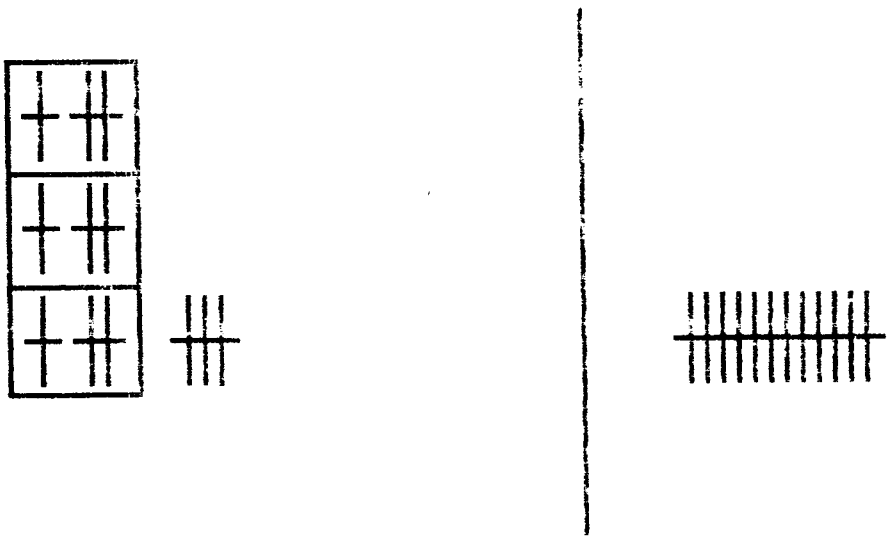
this goal is designed to allow the students to practice the STACK command



this goal represents a two step problem in which students must utilize both UNBUNDLE and STACK



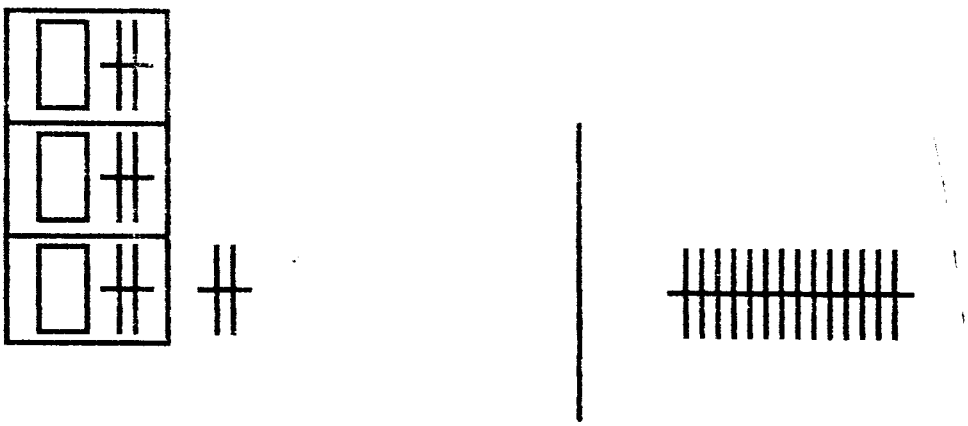
this goal represents a 3 step problem - students must use UNBUNDLE, STACK and UNBUNDLE again to achieve the goal



this goal represents a progression to solving for an unknown, or finding out what belongs in the box - it can be reached in one UNBUNDLE step

this goal represents a two step problem - students must use both UNBUNDLE and STACK to determine the contents of the box

this is the most difficult type of goal - students must use UNBUNDLE, STACK and UNBUNDLE again to find out what needs to be in the box



FUTURE DIRECTIONS

Although all instructional models are imperfect, there are some very obvious drawbacks to "Stacker" in its present state. First, the concepts of subtraction and negative numbers must be addressed. Algebraic manipulations involving subtraction and/or negative numbers are notorious sources of difficulty for students, and "Stacker" would be greatly improved by including representations of these concepts. Also, a linked representation, that is, including symbolic representations corresponding to the iconic "bundles" should theoretically increase transfer to a conventional classroom treatment of the equations modeled. Finally, the ability to represent more complicated equations such as $3(3x + 2) = 24$ (i.e. stacks within stacks) would increase the applicability of this program. Methods of incorporating each of these issues have been considered. Given additional time, these adaptations could be made with relatively little difficulty.

TECHNICAL DESCRIPTION

What follows is a brief overview, in outline form, of the procedures which comprise "Stacker".

Commands (procedures) entered by user:

```
TO GOAL :N
     - checks validity of :N
     - uses:
          TO GOAL1 :N
               - initiates goal state :N
               - sets values of :LEN and :GAP

TO UNDO
     - undoes last command
     - uses:
          TO DRAW :SIDE :LEN :GAP
               - graphically represents :SIDE on the computer
                 screen with stick length :LEN and distance between
       sticks :GAP

TO SO
     - reinitiates current goal state
     - uses:
          TO DRAW :SIDE :LEN :GAP
               (see above)

TO BUNDLE :TERM1 :TERM2
     - checks validity of :TERM1 and :TERM2
     - uses:
          TO CONVERT :CHAR
               - converts a letter of the alphabet (a-z) to a
                 number (1-26)
          TO BUNDLE1 :TERM1 :TERM2
               - alters the screen representation of :SIDE
                 by combining :TERM1 and :TERM2 and placing
                 the result where :TERM1 was
               - uses:
                    TO REPL :L :N :VAL
                         - replaces the :Nth word of list :L
                           with :VAL
                    TO DELETE :L :TERM
                         - deletes the :TERMth word of list :L
                    TO DRAW :SIDE :LEN :GAP
                         (see above)
```

```
TO UNBUNDLE :TERM :SIZE1
     - checks the validity of :TERM :SIZE1
     - uses:
          TO CONVERT :CHAR
               (see above)
          TO UNBUNDLE1 :TERM :SIZE1
               - separates term :TERM into two bundles of size :SIZE1
                 and what is left
               - uses:
                    TO UNBUNDLEST :TERM :SIZE1
                         - further checks validity of :TERM and
                           :SIZE1
                         - performs same operation as TO UNBUNDLE
                           only within a stack
                         - uses:
                              TO REPL2 :L :TERM :SIZE1 :SIZE2
                                   - replaces word :TERM in list :L
                                     with the two words :SIZE1 and
                                     :SIZE2
                              TO REPL :L :N :VAL
                                   (see above)
                    TO REPL2 :L :TERM :SIZE1 :SIZE2
                         (see above)
                    TO DRAW :SIDE :LEN :GAP
                         (see above)

TO STACK :TERM :HEIGHT :LINE
     - checks validity of :TERM :HEIGHT :LINE
     - uses:
          TO CONVERT :CHAR
               (see above)
          TO STACK1 :TERM :HEIGHT :LINE
               - alters screen representation changing a bundle to
                 a stack of height :HEIGHT with :LINE in each box
               - uses:
                    TO REPL :L :N :VAL
                         (see above)
                    TO DRAW :SIDE :LEN :GAP
                         (see above)
```

```
TO UNSTACK :TERM
     - checks validity of :TERM
     - uses:
          TO CONVERT :CHAR
               (see above)
          TO UNSTACK1 :TERM
               - changes a stack to a bundle the size of
                 each bundle of the stack added together
               - uses:
                    TO SUMT :L
                         - adds up all of the words in list :L
                    TO REPL :L :N :VAL
                         (see above)


Procedures used to create graphic display


TO DRAW :SIDE :LEN :GAP
     - uses:
          TO DOSIDE :SIDE :LEN :GAP
               - represents :SIDE on the screen with stick length
                 :LEN and distance between sticks :GAP
               - uses:
                    TO DOSTACK :S :LEN :GAP
                         - draws the stack represented by :S on the
                           screen
                         - uses:
                              TO DOINIT
                                   - initializes parameters used
                                     in TO DOBOX
                              TO DOLINE :L :LEN :GAP
                                   - draws a line of bundles repre-
                                     sented by :l
                                   - uses:
                                        TO DOX :LEN :GAP
                                             - draws a box with
                                               dimensions :LEN by
                                               3(:GAP) to represent
                                               an "X"
                                        TO DOBUNDLE :NUM :LEN :GAP
                                             - draws a bundle of :NUM
                                               sticks of length :LEN
                                               with distance :GAP
                                               between them
                                             - uses:
                                                  TO DOSTICK :LEN :GAP
                                                       - draws one
                                                         of length
                                                         :LEN
```

TO DOTIE :NUM :LEN
:GAP
- draws the tie
on the
bundle

TO DOBOX
- draws a box around a line of
bundles to form a stack
- uses:
TO DOPOSIT :WIDTHB :GAP
- repositions the turtle
to draw the next line
of the stack based on
the width of the box
and :GAP

TO DOXS :LEN :GAP
- draws a box to represent "X" outside
of a stack

TO DOBUNDLES :NUM :LEN :GAP
- draws a bundle of :NUM sticks of length
:LEN with distance between sticks :GAP
outside of a stack
- uses:
TO DOSTICK :LEN :GAP
- draws one stick of length :LEN
TO DOTIE :NUM :LEN :GAP
- draws the tie on the bundle

SELECTED BIBLIOGRAPHY

Carpenter, T. P., Corbitt, M. K., Kepner, H. S., Lindquist, M. M., & Reys, R. E. (1981). Variables and relationships. In Results from the Second International Mathematical Assessment of the NAEP (pp. 56-71). Reston, VA: NCTM.

Dede, C. J. (1987, November). Empowering environments, hypermedia and microworlds. The Computing Teacher, 20-24.

Herscovics, N. & Kieran, C. (1980, November). Constructing meaning for the concept of equation. Mathematics Teacher, 73, 572-580.

Martin, W. G. (1987). Computer Geometry: Logo and Beyond. Unpublished Manuscript.

Papert, S. (1980). Mindstorms: Children, Computers and Powerful Ideas.

Thompson, P. W. (1985, September). A Pigetian approach to transformation geometry. Mathematics Teacher, 78, 465-471.

```
 POPS
TO GOAL1 :N
LOCAL "TSIDE
MAKE "LEN 20
MAKE "GAP 3
SETSG 6
CLEAR
CATCH "ERROR [ERF "TARGET]
MAKE "L ITEM :N :GOALS
MAKE "TSIDE ITEM 1 :L
MAKE "SIDE ITEM 2 :L
MAKE "SSIDE :SIDE
LOADPIC "LINE
PU
SETPOS [-100 -60]
DRAW :TSIDE :LEN :GAP
SAVEPIC "TARGET
PU
SETPOS [20 -60]
DRAW :SIDE :LEN :GAP
END

TO H
OP "H
END

TO G
OP "G
END

TO F
OP "F
END

TO E
OP "E
END

TO D
OP "D
END

TO C
OP "C
END

TO B
OP "B
END

TO UNDO
MAKE "SIDE :OLDSIDE
CLEAR
LOADPIC "TARGET
PU
SETPOS [20 -60]
DRAW :SIDE :LEN :GAP
END

TO DELETE :L :TERM
IF :TERM = 1 [MAKE "L BF :L] [MAKE "L FPUT FIRST :L DELETE BF :L :TERM - 1]
```

```
OP :L
END

TO BUNDLE1 :TERM1 :TERM2
LOCAL [VAL1 VAL2]
MAKE "VAL1 ITEM :TERM1 :SIDE
MAKE "VAL2 ITEM :TERM2 :SIDE
[illegible]E "OLDSIDE :SIDE
MAKE "SIDE REPL :SIDE :TERM1 :VAL1 + :VAL2
MAKE "SIDE DELETE :SIDE :TERM2
CLEAR
LOADPIC "TARGET
SETPOS [20 -60]
DRAW :SIDE :LEN :GAP
END

TO BUNDLE :TERM1 :TERM2
MAKE "TERM1 CONVERT :TERM1
MAKE "TERM2 CONVERT :TERM2
IF :TERM1 > COUNT :SIDE [( PR [TERM] CHAR ( :TERM1 + 64 ) [DOESN'T EXIST ) THROW "TOPLEVEL]]
IF :TERM2 > COUNT :SIDE [( PR [TERM] CHAR ( :TERM2 + 64 ) [DOESN'T EXIST] ) THROW "TOPLEVEL]
IF LISTP ITEM :TERM1 :SIDE [( PR [TERM] CHAR ( :TERM1 + 64 ) [CAN'T BE BUNDLED] ) THROW "TOPL
IF LISTP ITEM :TERM2 :SIDE [( PR [TERM] CHAR ( :TERM2 + 64 ) [CAN'T BE BUNDLED] ) THROW "TOPL
BUNDLE1 :TERM1 :TERM2
END

TO DOBUNDLES :NUM :LEN :GAP
LOCAL [X Y]
MAKE "X XCOR
MAKE "Y YCOR
PU
SETY :YPERM
REPEAT :NUM [DOSTICK :LEN :GAP]
[illegible]TIE :NUM :LEN :GAP

SETPOS LIST :X :Y
SETHEADING 0
RT 90
FD ( :NUM + 3 ) * :GAP
SETHEADING 0
END

TO DOXS :LEN :GAP
PD
FD :LEN
RT 90
FD 3 * :GAP
RT 90
FD :LEN
RT 90
FD 3 * :GAP
RT 90
MAKE "NUM 3
PU
RT 90
FD ( :NUM + 3 ) * :GAP
RT -90
END

TO DOSIDE :SIDE :LEN :GAP
[illegible]AL "FSIDE
IF :SIDE = [] [STOP]
MAKE "FSIDE FIRST :SIDE
[illegible]:GAP]:FSIDE [DOSTACK :FSIDE :LEN :GAP] [IF :FSIDE = "X [DOXS :LEN :GAP] [DOBUNDLES :FSIDE :L
DOSIDE BF :SIDE :LEN :GAP
END

TO DOSTACK :S :LEN :GAP
```

```
LOCAL [XLLC XLRC YLLC YLRC FS LENGTHB]
IF :S = [] [STOP]
PU
SETY :YPERM
MAKE "FS FIRST :S
REPEAT :FS [DOINIT DOLINE BF :S :LEN :GAP DOBOX]
SETY :YPERM
RT 90
FD ( :LENGTHB + :GAP )
LT 90
END

TO DOINIT
MAKE "XLLC XCOR - 2 * :GAP
MAKE "YLLC YCOR - :GAP
END

TO DOLINE :L :LEN :GAP
IF :L = [] [STOP]
MAKE "NUM FIRST :L
IF :NUM = "X [DOX :LEN :GAP] [DOBUNDLE :NUM :LEN :GAP]
RT 90
PU
FD ( :NUM + 2 ) * :GAP
RT -90
DOLINE BF :L :LEN :GAP
END

TO DOBOX
LOCAL [WIDTHB]
MAKE "XLRC XCOR
MAKE "YLRC YCOR
MAKE "LENGTHB ( ABS :XLRC - :XLLC ) - :GAP
MAKE "WIDTHB :LEN + 2 * :GAP
PU
SETPOS LIST :XLLC :YLLC
SETH 0
PD
FD :WIDTHB
RT 90
FD :LENGTHB
RT 90
FD :WIDTHB
RT 90
FD :LENGTHB
SETH 0
DOPOSIT :WIDTHB :GAP
END

TO DOX :LEN :GAP
PD
FD :LEN
RT 90
FD 3 * :GAP
RT 90
FD :LEN
RT 90
FD 3 * :GAP
RT 90
MAKE "NUM 3
END

TO DOTIE :NUM :LEN :GAP
IF :NUM = 0 [STOP]
SETHEADING 0
PU
```

```
FD :LEN / 2
RT 90
PD
FD ( :NUM + 1 ) * :GAP
END

TO DOSTICK :LEN :GAP
SETHEADING 0
FD :LEN
FD - :LEN
PU
RT 90
FD :GAP
END

TO DOBUNDLE :NUM :LEN :GAP
LOCAL [X Y]
MAKE "X XCOR
MAKE "Y YCOR
REPEAT :NUM [DOSTICK :LEN :GAP]
DOTIE :NUM :LEN :GAP
PU
SETPOS LIST :X :Y
SETHEADING 0
END

TO DOPOSIT :WIDTHB :GAP
PU
RT 90
FD :GAP * 2
LT 90
FD :WIDTHB + :GAP


TO REPL2 :L :TERM :SIZE1 :SIZE2
IF-:TERM=:SIZE1 [MAKE "L FPUT :SIZE1 ( FPUT :SIZE2 BF :L )] [MAKE "L FPUT FIRST :L REPL2 BF :L :T
OP :L
END

TO UNBUNDLEST :TERM :SIZE1
PR 1
MAKE "LIST ITEM :TERM :SIDE
PR 2
IF ( COUNT :LIST ) > 2 [( PR CHAR ( :TERM + 64 ) [CAN'T BE UNBUNDLED] ) THROW "TOPLEVEL]
PR 3
IF :SIZE1 > ITEM 2 :LIST [PR [SIZE IS TOO LARGE] THROW "TOPLEVEL]
PR 4
MAKE "NEWTERM REPL2 :LIST 2 :SIZE1 ( ITEM 2 :LIST - :SIZE1 )
PR 5
MAKE "SIDE REPL :SIDE :TERM :NEWTERM
PR 6
OP :SIDE
END

TO UNBUNDLE1 :TERM :SIZE1
MAKE "OLDSIDE :SIDE
IF=:SIDE [SETPOS [0 0] MAKE "SIDE [] THROW "TOPLEVEL]
UNBUNDLEST :TERM :SIZE1
CLEAR
LOADPIC "TARGET
SETPOS [20 -60]
DRAW :SIDE :LEN :GAP
END

TO UNBUNDLE :TERM :SIZE1
LOCAL [TEST]
```

```
MAKE "TERM CONVERT :TERM
IF NOT NUMBERP :SIZE1 [PR [SIZE MUST BE A NUMBER] THROW "TOPLEVEL]
MAKE "TEST ( AND :SIZE1 > 0 :SIZE1 = INT :SIZE1 )
IF :TEST = "FALSE [PR [BAD INPUT TO UNBUNDLE] THROW "TOPLEVEL]
IF :TERM > COUNT :SIDE [( PR [TERM] CHAR ( :TERM + 64 ) [DOESN'T EXIST] ) THROW "TOPLEVEL]
UNBUNDLE1 :TERM :SIZE1
END

TO STACK1 :TERM :HEIGHT :LINE
PR 1
MAKE "OLDSIDE :SIDE
MAKE "SIDE REPL :SIDE :TERM LIST :HEIGHT :LINE
CLEAR
LOADPIC "TARGET
SETPOS [20 -60]
DRAW :SIDE :LEN :GAP
END

TO STACK :TERM :HEIGHT :LINE
MAKE "TERM CONVERT :TERM
PR 1
IF :TERM > COUNT :SIDE [( PR [TERM] CHAR ( :TERM + 64 ) [DOES NOT EXIST] ) THROW "TOPLEVEL]
PR 2
PR :TERM
IF LISTP ITEM :TERM :SIDE [( PR [TERM] CHAR ( :TERM + 64 ) [IS ALREADY A STACK] ) THROW "TOPL
PR 3
IF NOT :LINE = ITEM :TERM :SIDE [( PR [I CAN'T STACK] CHAR ( :TERM + 64 ) :HEIGHT
PR 4
STACK1 :TERM :HEIGHT :LINE
END

TO REPL :L :N :VAL
IF :N = 1 [MAKE "L FPUT :VAL BF :L] [MAKE "L FPUT FIRST :L REPL BF :L :N - 1 :VAL]
OP :L
END

TO SUMT :L
IF :L = [] [OP 0] [OP ( FIRST :L ) + SUMT BF :L]
END

TO UNSTACK1 :TERM
LOCAL "NEWTERM
LOCAL ["VAL "NEWTERM]
MAKE "VAL 0
MAKE "L ITEM :TERM :SIDE
MAKE "NEWTERM ( SUMT BF :L ) * FIRST :L
MAKE "OLDSIDE :SIDE
MAKE "SIDE REPL :SIDE :TERM :NEWTERM
CLEAR
LOADPIC "TARGET
SETPOS [20 -60]
DRAW :SIDE :LEN :GAP
END

TO CONVERT :CHAR
LOCAL "TEST
MAKE "N ASCII :CHAR - 64
MAKE "TEST ( AND :N > 0 :N < 27 :N = INT :N )
IF :TEST = "TRUE [OP :N] [PR [BAD INPUT - - TERMS ARE LABLED A - Z] THROW "TOPLEVEL]
END

TO UNSTACK :TERM
MAKE "TERM CONVERT :TERM
IF :TERM > COUNT :SIDE [( PR [TERM] CHAR ( :TERM + 64 ) [DOESN'T EXIST] ) THROW "TOPLEVEL]
IF NOT LISTP ITEM :TERM :SIDE [( PR [TERM] CHAR ( :TERM + 64 ) [CAN'T BE UNSTACKED] ) THROW
UNSTACK1 :TERM
END
```

```
TO DRAW :SIDE :LEN :GAP
LOCAL "NUM
MAKE "YPERM -60
DOSIDE :SIDE :LEN :GAP
SETY :YPERM
PU
END

TO CLEAR
CS
END

TO SO
CLEAR
LOADPIC "TARGET
PU
SETPOS [20 -60]
DRAW :SSIDE :LEN :GAP
END

TO GOAL :N
LOCAL "TEST
MAKE "TEST ( AND NUMBERP :N :N > 0 :N = INT :N )
IF :TEST = "FALSE [PR [BAD INPUT TO GOAL] THROW "TOPLEVEL]
IF :N > 44 [PR [THERE ARE ONLY 44 GOALS AVAILABLE] THROW "TOPLEVEL]
GOAL1 :N
END

TO TR :N
IF :N > COUNT :GOALS [STOP]
GOAL :N
TYPE RC
TR :N + 1
END

TO a
OP "A
END


?
```